Description

GENERIC DETECTION AND ELIMINATION OF MACRO VIRUSES

Technical Field

This invention pertains to the field of detecting and eliminating computer viruses of a particular class known as macro viruses.

Background Art

U.S. patent 5,398,196 discusses the detection of viruses within a personal computer. However, unlike the present invention, this reference does not treat the elimination of detected viruses, nor does it discuss macro viruses.

A first existing technology used by anti-virus programs to detect and repair macro viruses requires, for each unique new macro virus, the development of a detection and repair definition (virus signature). Thus, this first technology is capable of detecting only publicly identified macro viruses (see below definition). After the development of the detection and repair definition, the user's anti-virus software must be augmented with the new definition before it can detect the newly discovered macro virus. This method has the advantage that a skilled anti-virus researcher is able to study the virus and understand it enough so that a

proper detection and repair definition can be created for it. The main disadvantage is that a relatively long turnaround time is required before the general public is updated with each new definition. The turnaround time includes the duration during which the virus has a chance to spread and possibly wreak havoc, the time to properly gather a sample and send it to an anti-virus research center, the time required to develop the definition, and the time to distribute the definition to the general public. This process is similar to the process used for protecting against the once more prevalent DOS-based viruses.

A second technology uses rudimentary heuristics that can scan for newly developed macro viruses. These heuristics employ expert knowledge of the types of viruses they seek. Often these heuristics look for strings of bytes that are indicative of viral behavior--for example, strings found in currently known viruses. Current heuristics are very good at detecting, with high level of confidence, new viruses that are variants of known viruses, but not so good at detecting new viruses that are not variants of known viruses. Another disadvantage of most current heuristics is that they are good for viral detection only, and are not capable of eliminating macro viruses once found. This is true of both macro virus heuristics and DOS virus heuristics. The present invention is an example of the second technology (heuristics). It is capable

of identifying both publicly identified macro viruses and publicly unidentified macro viruses (see the below definitions). Furthermore, it offers the significant advantage that it is capable of eliminating macro viruses that are detected.

Disclosure of Invention

A computer implemented method, apparatus, and computer readable medium for detecting macro viruses in code (15) adapted for use on a digital computer (1). A detection module (17) analyzes the code (15) to determine whether the code (15) contains instructions causing a macro (8) to be moved to a global environment (13), and whether the code (15) also contains instructions causing the same macro (8) to be copied to a local document (11). When these two conditions are satisfied, detection module (17) declares that a macro virus is present within the code (15).

Brief Description of the Drawings

These and other more detailed and specific objects and features of the present invention are more fully disclosed in the following specification, reference being had to the accompanying drawings, in which:

Figure 1 is a block diagram showing the type of application program 5 in the existing body of art that can be contaminated by macro viruses detectable by the present invention.

Figure 2 is a block diagram showing global environment 13 associated with application program 5 of Figure 1.

Figure 3 is a block diagram showing how a macro virus can contaminate the computing environment illustrated in Figures 1 and 2.

Figure 4 is a block diagram showing a preferred embodiment of the present invention.

Figure 5 is a flow diagram showing a preferred embodiment of the present invention.

Definitions

As used throughout the present specification and claims, the following words and expressions have the indicated meanings:

"macro" is a computer program written using a structured programming language and created from within an application program that has a global environment and can create local documents. Normally, a macro can be invoked using a simple command such as a keystroke. The application program can be, for example, Microsoft Word or Microsoft Excel.

"global environment" is an area within a storage medium that is associated with a particular application program and stores parameters and/or macros that are used every time said application program is used. For example, the global environment for a particular application program can contain text, graphics, and one or more macros.

"local document" is a document that has been generated by an application program.

"virus" is a malicious computer program that replicates itself.

"macro virus" is a virus consisting of one or more macros.

"payload" is an unwanted destructive task performed by a virus. For example, the payload can be reformatting a hard disk, placing unwanted messages into each document created by an application program, etc.

"heuristics" is a set of inexact procedures.

"publicly identified macro virus" is a macro virus that has a known viral signature.

"publicly unidentified macro virus" is a macro virus that cannot be identified by antivirus software using viral signature matching techniques.

Detailed Description of the Preferred Embodiments

The purpose of the present invention is to detect and eliminate macro viruses in a generic manner, i.e., the present invention works regardless of whether the virus is a publicly identified or a publicly unidentified macro virus.

The present invention uses heuristics that can determine effectively whether any given code 15 contains a macro virus or not, and determine exactly the set of macros that comprise the virus. This is achieved by means of detection module 17 analyzing code 15 for the presence of two types of instructions, as described below.

The present invention offers the following advantages over the prior art:

- a generic detection and repair solution for new (publicly unidentified) macro viruses.
- virtually no turnaround time.

- ability to determine with an extremely high degree of confidence that a set of macros declared to be a virus is indeed a virus.

- ability to determine the set of macros that comprise the virus, thus providing an immediate repair solution.

- reduced workload for all personnel involved, in terms of virus discovery, analysis, and definition creation.

- increased user satisfaction with regard to protection against new (publicly unidentified) viruses.

Figure 1 illustrates a typical operating environment of the present invention. A digital computer 1 comprises a processor 4 and memory 3. When it is to be executed, application program 5 is moved into memory 3 and is operated upon by processor 4. Application program 5 is any program that generates macros, for example, Microsoft Word or Microsoft Excel. When it is executed, application program 5 generates one or more local documents 11, which are stored in storage medium or media 9 associated with computer 1. For example, storage medium 9 can be a hard disk, floppy disk, tape, optical disk, or any other storage medium used in connection with digital computers. Each document 11 can comprise text, graphics, and/or one or more macros 8. Each macro 8 is typically contained within a module 12. A module can contain

one or more macros 8. A user of computer 1 typically communicates with application program 5 via user interface 7, which may comprise a keyboard, monitor, and/or mouse.

Figure 2 shows a document 11 that has been opened by application program 5. Because document 11 has been so opened, it resides in memory 3, where it can be readily and quickly accessed by application program 5. As stated previously, document 11 can contain one or more macros 8. If one of these macros 8 is named AutoOpen or a similar name, said macro 8 will execute automatically. Alternatively, the macro 8 could execute upon the user pressing a certain key on keyboard 7, or upon the occurrence of another event.

Figure 2 also illustrates the presence of the global environment 13 that is associated with application program 5. Global environment 13 is available to the user every time he or she uses application program 5, and is specific to each such application program 5. Global environment 13 typically contains a set of macros 8 previously established by the user, orders of menus, new menu items, and preferences of the user, e.g., font styles and sizes.

There is typically just one global environment 13 per computer 1 per application program 5, even if different users share the same

program 5. Global environment 13 is located within storage medium 10. Storage medium 10 can be the same storage medium 9 as used by one or more documents 11 that have been generated by application program 5. Alternatively, storage medium 10 may be distinct from storage medium 9 or storage media 9. Storage medium 10 can be any storage device used in conjunction with a digital computer, such as a hard disk, floppy disk, tape, optical disk, etc.

If application program 5 is the spreadsheet software known as Microsoft Excel, then global environment 13 has a typical location which is the subdirectory named "XLSTART" in the directory where Excel is stored (i.e., "C:/EXCEL/XLSTART"); a document 11 is called a "workbook"; macros 8 reside within modules 12 in the workbook; and the language in which the macros 8 are written is Visual Basic, an object oriented language.

Figure 3 illustrates how macro viruses propagate (replicate) into the global environment 13. In step 1, document 11 is opened by application program 5. During step 1, document 11, including all the elements contained therewithin, move from storage medium 9 to memory 3. In the illustrated embodiment, document 11 comprises a module 12 containing a macro 8, text, and graphics. The text may be, for example, a letter that the user has created previously. All of these items move to memory 3.

Let us assume that macro 8 contains a virus. In step 2, in the case where program 5 is Microsoft Excel, macro virus 8 causes the entire document 11 to be moved to global environment 13.

In step 3, macro virus 8 manifests its payload, usually by copying itself into another local document 11. Step 3 can be precipitated every time a new document 11 is generated by application program 5 or less often; for example, every time that document 11 is a letter addressed to a certain individual. In any event, the payload of macro 8 has a highly negative effect on computer 1. For example, this payload can entail infecting certain documents 11 with gibberish, reformatting a storage medium 9, 10, etc.

Thus does macro virus 8 infect the global environment 13, and from there is poised like a coiled snake ready to infect other documents 11. This is because the global environment 13 is always active, and thus, macro virus 8 is always active. From the newly infected documents 11, this virus 8 can infect the global environments 13 of all users to whom the infected documents 11 are passed.

Figure 4 illustrates apparatus by which the present invention detects and eliminates macro viruses. Code 15 is what is analyzed by the present invention for the presence of macro viruses. Code

15 is adapted to be used from within computer 1, where it will be coupled to the documents 11 generated by application program 5 and to global environment 13. Coupled to code 15 is detection module 17, which is the module of the present invention that determines whether a macro virus is present within code 15, based upon the simultaneous presence of two conditions, as described below. Module 17 can be implemented in hardware, firmware, and/or software.

Figure 4 illustrates the special case where detection module 17 and code 15 are simultaneously resident within computer 1. However, code 15 does not have to be resident within computer 1 in order for the present invention to work; code 15 can be resident on a hard disk, floppy disk, or other storage medium not currently a part of computer 1.

Detection module 17 is coupled to user interface 7, so that module 17 may announce its decisions concerning detection of macro viruses to the user. Coupled to detection module 17 is repair module 19, which eliminates macro viruses that have been determined by detection module 17 to be present. Repair module 19 is also coupled to code 15. Module 19 can be implemented in hardware, firmware, and/or software.

The method of the present invention is illustrated in Figure 5. In step 51, detection module 17 begins to analyze code 15. It does this by examining a first module 12 within code 15. Module 17 first determines whether the module 12 contains a macro 8. The determination as to whether something is a macro is a conventional technique in computer programming. Another (optional) preprocessing step is for module 17 to determine the language in which code 15 has been written. This determination is again a conventional technique of computer programming. In the special case where code 15 is meant to be used in association with the Microsoft Excel spreadsheet application program 5, this step is not necessary, because it is known that the macros 8 in Excel have to be written in the Visual Basic language.

In step 52, module 17 determines whether the code 15 under test contains instructions that cause a macro 8 to be moved to a global environment 13. This is the first of two conditions that module 17 uses to determine the presence of a macro virus. In the case where the code 15 is written in Visual Basic, module 17 preferably deems that this first condition is satisfied when a SaveAs command that performs this operation is present within code 15. If module 17 determines that the first condition is not satisfied, module 17 declares at step 57 that no macro virus is present within the analyzed macro 8. This declaration may be

passed to the user via user interface 7. At this point, there is no need for module 17 to continue analyzing that macro 8, so at step 56 module 17 goes to the next macro 8, and analyzes this new macro 8 as before.

If, on the other hand, the result of step 52 is that module 17 has determined that the first condition for the presence of a macro virus has been satisfied, module 17 flags the macro 8 under test, and proceeds to the next step, step 53, where module 17 determines whether the second condition for the presence of a macro virus is satisfied. This condition is the presence within module 12 of instructions that copy the same macro 8 that was flagged within step 52 to a local document 11. In the case where the code 15 is written in the Visual Basic language, the determination as to whether a macro 8 is being copied to a local document 11 is preferably made by detecting the presence of a Copy command that performs this operation. If this second condition is not satisfied, module 17 again declares at step 57 that no macro virus is present.

If, on the other hand, module 17 has determined within step 53 that the second condition for the presence of a macro virus has been satisfied, both conditions have now been satisfied, and module 17 declares at step 54 that a macro virus is indeed present within

module 12. This declaration can be passed to the user via user interface 7. As this point, in the preferred embodiment, repair module 19 eliminates the virus at step 55. This is typically done by "deleting" the virus, as that term is used in conventional computer programming. In the preferred embodiment, repair module 19 deletes the entire module 12 containing the virus. Alternatively, module 19 deletes just the macro 8 that was flagged in step 52 as the subject of a move to a global environment 13 and in step 53 to be the subject of a copy to a local document 11.

Lets us now give an example to illustrate the operation of the preferred embodiment. In this example, application 5 is Microsoft Excel and module 12 comprises three lines of Visual Basic code as follows:

```
start = Application.StartupPath
newname = ActiveWorkbook.Name
Workbooks(newname).SaveAs filename:=start & "/" &
    "main.xls"
```

The first two lines of code constitute the initialization of variables named "start" and "newname", respectively. The third line of code is a command line. "Workbooks" is the set of all workbooks that are currently open. Initially it is a null set, because no documents 11 are open the first time Excel is opened. The presence of the SaveAs command means that something called Workbooks(newname) is being saved someplace. "filename" is a

parameter to the SaveAs command. We know that is a parameter because of the ":=" following it. The item being saved is saved to the location "start/main.xls". Note the presence of the two concatenation operators "&". The presence of these concatenation operators may well indicate that the author of the virus attempted to disguise the presence of the virus by breaking up the destination location of the SaveAs into three parts. "Application.StartupPath" evaluates to Excel's startup directory, which is its global environment 13. Assuming that this location is "C:/EXCEL/XLSTART", when we substitute the value of "Application.StartupPath" for the variable "start", we see that the item to be saved is saved in the location "C:/EXCEL/XLSTART/main.xls". main.xls is an arbitrary name given by the author of the virus. We know that Application.StartupPath is the location of the startup directory within a global environment 13. Global environment 13 also contains an alternative startup directory that can be referred to from within a Visual Basic macro using Application.AltStartupPath.

Making the substitution of variables, the item being saved to the global environment 13 is a local document 11 called "ActiveWorkbook.Name". In the context of this module 12, this active workbook 11 is the one containing the module 12, and thus

this module 12 (which is known to contain a macro 8) is saved to the global environment 13.

Thus, module 12 contains instructions causing a macro 8 to be moved to a global environment 13. Therefore, the first condition (step 52) for the presence of a macro virus is satisfied.

In our example, let us assume that module 12 also contains the following code:

```
papy_name = ActiveWorkbook.Name
Workbooks("MAIN.XLS").Sheets("junior").Copy
   before:=Workbooks(papy_name).Sheets(1)
```

In the Visual Basic syntax, something to the left of an "=" is a variable.

"Name" is a user-given name.

"junior" is the name of the module 12 that contains the virus. We know that the module 12 containing the three lines of code that satisfied condition one is in "junior" (part of main.xls), because we assume that module 17 has noted that "junior" is the name of the module 12 containing this code. Alternatively, we can use other means to determine where the suspicious code 15 is going to be sent, e.g., we can determine whether there is some other code renaming "junior" as "senior".

In this example, "before" means "the location just before". Here the syntax requires an "after" or a "before". In the second line of this code, the module named "junior" from the globally installed workbook 11 named "main.xls" is copied to the active workbook 11, which, at the time of execution of the above two lines, is the workbook 11 currently being edited by the user.

Thus, the same module 12 that was flagged in step 52 as being saved into the global environment 13 is now uncovered to be copied into a local document (workbook 11). Therefore, module 17 determines that condition two is satisfied, and therefore declares that this module 12 contains a macro virus.

In order for the implementation of this invention to be robust, and thus able to detect the greatest number of viruses, it is desirable, although not necessary, for detection module 17 to have one or more of the following four enhancement features. Each of these features slows down the operation of module 17. However, one can implement each of these features with extreme robustness or minimum robustness, depending upon the degree of robustness expected to be desirable. The factors that go into this determination include memory 3 constraints, execution time constraints, and expected real world scenarios.

The four enhancement features are:

1. The ability to understand and handle string concatenation operations in the analyzed modules 12. For example, any string in Visual Basic can be concatenated. The reason for this first enhancement feature is that, for example, the virus designer could have referred to "junior" as "jun" & "ior". In this case, it is desirable for module 17 to recognize this as "junior". Module 17 will so recognize this as "junior" if enhancement feature 1 is present.

2. The second enhancement feature is the ability for module 17 to understand and handle variable assignment operations in order to trace the values of variables. This is sometimes referred to as variables being "proxied". For example, the virus designer could have written the code to be:

```
papy_name=ActiveWorkbook.name
a="jun"
b="ior"
c=a&b
Workbooks("MAIN.XLS").Sheets(c).Copy
    before:=Workbooks(papy_name).Sheets(1)
```

If we want to detect "junior" in this case, we need to make all of these variable substitutions. Since there is a constraint on the size of memory 3, and on the amount of time that can be used, we might limit module 17 to where it will perform only an

arbitrary number (for example 16) of substitutions. This is because it is expected that the amount of variable usage is small in most cases, and we accept this tradeoff between robustness and speed/memory constraints.

3. The third enhancement feature is for module 17 to have the ability to trace the value of parameter values through function calls (subroutine calls). For example, suppose that code 15 contains:

```
sub CopyIt(SheetName)
      papy_name = ActiveWorkbook.Name
      Workbooks("MAIN.XLS").Sheets(SheetName).Copy
        before:=workbooks(papy_name)Sheets(1)
End Sub

Sub InfectDocument
   sName = "junior"
   CopyIt(sName)
End Sub
```

In this example, the value of the parameter variable SheetName is being passed to the subroutine CopyIt from another subroutine (function) called InfectDocument, in an attempt by the virus designer to obfuscate what is happening.

4. The fourth enhancement feature is giving module 17 the ability to understand and handle the object model of Visual Basic, e.g., object/subobject hierarchy parsing and understanding. For

example, a straightforward way of initializing the variable "start" would be to write:

```
start=Application.StartupPath
```

However, if the virus designer wanted to obfuscate what he or she was doing, said designer might write:

```
app=Application
start=app.StartupPath
```

In this case, "Application" is an object. If the fourth enhancement feature is present, module 17 has the ability to handle substituted object names.

The above description is included to illustrate the operation of the preferred embodiments and it is not meant to limit the scope of the invention. The scope of the invention is to be limited only by the following claims. From the above discussion, many variations will be apparent to one skilled in the art that would yet be encompassed by the spirit and scope of the present invention.

What is claimed is: